Comparing formal verification approaches of interlocking systems

Haxthausen, Anne Elisabeth; Nguyen, Hoang Nga; Roggenbach, Markus

Published in:
Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification

Link to article, DOI:
10.1007/978-3-319-33951-1_12

Publication date:
2016

Document Version
Peer reviewed version

Comparing formal verification approaches of interlocking systems

Anne Elisabeth Haxthausen[1], Hoang Nga Nguyen[2], and Markus Roggenbach[3]

[1] DTU Compute, Technical University of Denmark, Denmark,
[2] Centre for Mobility and Transport, Coventry University, UK
[3] Swansea Railway Verification Group, Swansea University, Wales, UK

Abstract. The verification of railway interlocking systems is a challenging task, and therefore several research groups have suggested to improve this task by using formal methods, but they use different modelling and verification approaches. To advance this research, there is a need to compare these approaches. As a first step towards this, in this paper we suggest a way to compare different formal approaches for verifying designs of route-based interlocking systems and we demonstrate it on modelling and verification approaches developed within the research groups at DTU/Bremen and at Surrey/Swansea. The focus is on designs that are specified by so-called control tables. The paper can serve as a starting point for further comparative studies.

1 Introduction

An interlocking system is responsible for guiding trains safely through a given railway network. It is a vital part of any railway signalling system and has the highest safety integrity level (SIL4) according to the CENELEC 50128 standard [3].

Conventionally, the development and verification process of interlocking systems is informal and mostly manual.

Adding automated verification. The left-hand picture in Figure 1 provides some detail as to how a conventional design process of interlocking systems is typically realised. Concretely it shows the process as implemented by our industrial partner Siemens Rail Automation, UK, in the form of a UML activity diagram. The client provides a CAD plan of the track plan and routes. Independently, the regulator provides a set of design rules. Based on these, the routes are signalled, i.e., various tables are developed. This scheme plan (i.e., track plan plus various tables, e.g., control tables) undergoes thorough manual checks before the tables are used to implement an interlocking. These checks are part of quality control: motivated on the one hand to detect mistakes early, already in the design phase, on the other hand to adhere to development standards required by the authorities as part of a certification process.

As the manual checks are time-consuming, costly, and error-prone, automated verification of interlocking systems is an active research topic. The right-hand

picture in Figure 1 shows a lightweight integration of automated verification (AV) into the traditional work-flow. It includes an automated check of the scheme plan for safety conditions. Only if a scheme plan has been proven to be safe, the costly manual checks are performed. Here, we deliberately refrain from replacing the manual checks. One reason is that safety covers only part of them. Furthermore, academic tools often have not been certified to the tool qualification levels required in safety cases. Finally, the railway domain is conservative: replacing traditional checks by different methods would require additional arguments in safety cases.

Automated verification of interlocking designs. It is still an open research question as how to perform safety checks on interlocking designs. The challenge is how to cope with the complexity of the problem: the state space grows exponentially in the size of the scheme plan to be verified. Several research groups, see e.g. [11, 1, 9, 6, 8, 7, 27, 16, 15, 13, 12, 4, 10, 29, 28, 23, 2, 20], have been addressing this challenge and have developed a number of different modelling and verification approaches.

The modelling part of such approaches usually consists of "transformations" of how to derive a (formal) model from informal rail descriptions as used in rail industry such as a track plan (e.g., as a CAD drawing) enriched by various tables (e.g., a control table). Similarly, the verification part usually states a safety condition (e.g., no train collision) and expresses this as a (formal) property (e.g., as a logical formula). Finally, an (automated) verification tool is utilised to provide an answer if the property holds in the model.

Different groups apply different design rules for signalling, country specific rail standards, utilising various modelling languages, employing different verification tools. This leads to the natural, however, fundamental question: how can these many modelling and verification approaches be related with each other? This question comes in at least three, interconnected forms: (i) how to relate the input of these modelling approaches? (ii) how to relate the formal models? (iii) how to relate the verification results?

Relating automated verification approaches. In this paper we suggest a general way of how to compare different formal verification approaches for interlocking systems and demonstrate it on modelling and verification approaches developed within our respective research groups at DTU/Bremen and at Surrey/Swansea. We see this comparison as pioneering work, to which we hope – in the long run – other groups will contribute as well by running their verification approaches through the very same exercises, i.e., this paper can serve as a start for a benchmark for railway verification.

While the focus of our comparison is on verification, to a certain extent we also address the other two questions posed above. Concerning input, we define a common core – see Section 2 – and discuss group specific extensions. Concerning the models, for a start we attempt to present the modelling approaches in a uniform way – see Section 3. The development of a general questionnaire on models will require more experience, including analysing work by further groups.

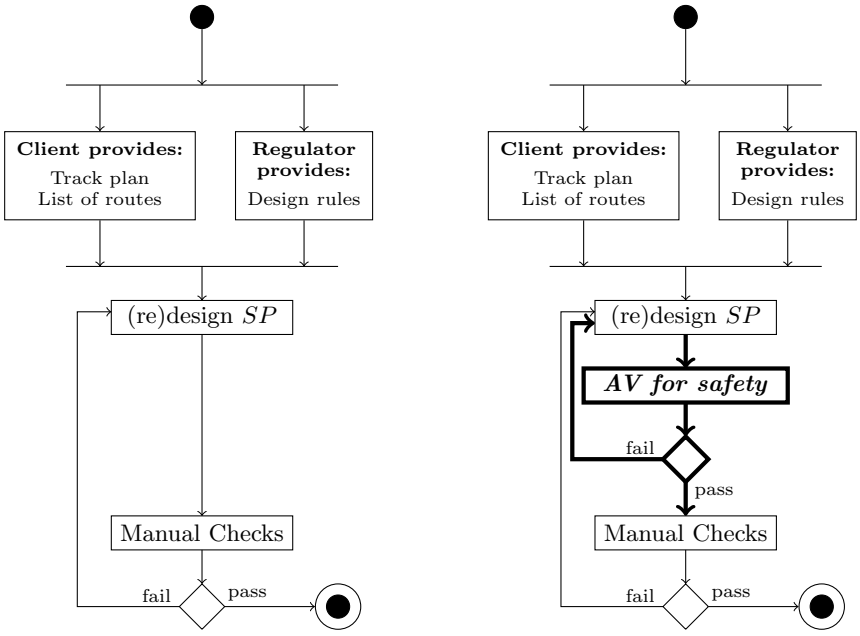


Fig. 1. Incorporating automated verification (AV) in a design & check cycle for a scheme plan SP.

Concerning relating verification, we are confident to propose a practicable and useful approach.

To the best of our knowledge, we are the first to address the question of how different verification approaches for interlocking systems relate. This question is important for various reasons: to advance the field, it will be necessary to better understand work from other groups and to learn from each other; from a company's point of view it is important to be able to evaluate different techniques when choosing one to be included in an interlocking design cycle.

Our comparison focuses on the safety claims that different approaches make. Rather than directly comparing models or properties, we look into the ability of verification approaches to detect errors in the design tables of an interlocking system. Starting with a correct rail design, we inject an error (e.g., by altering the entry of a table), and see if – under a given modelling and verification approach – the injected error is caught. As rail designs can be fault tolerant, here it is necessary to work with "minimal" designs, i.e., such designs, where an injected error actually can be caught. When comparing now two verification approaches, we say that an approach has more distinguishing power than another one, if the first flags the same errors as the second and possibly more.

Our comparison draws on ideas from testing theory. Yu and Lau prove a number of theorems on the error detecting capabilities of various coverage criteria

for testing logical decisions [30]. Their set-up consists of several elements: the logical decision to be tested needs to be in a normal form – corresponding to our minimal designs, see Section 2; they define a number of syntactic errors (e.g., adding or forgetting a negation, confusing a logical and with a logical or) – we will define a number of error types, see Section 4; they establish theorems that characterise the kind of errors that will be detected via testing provided a test suite has been constructed to minimally fulfil a certain coverage criterion – we will compare two verification approaches by defining and performing a number of experiments, see Section 5.

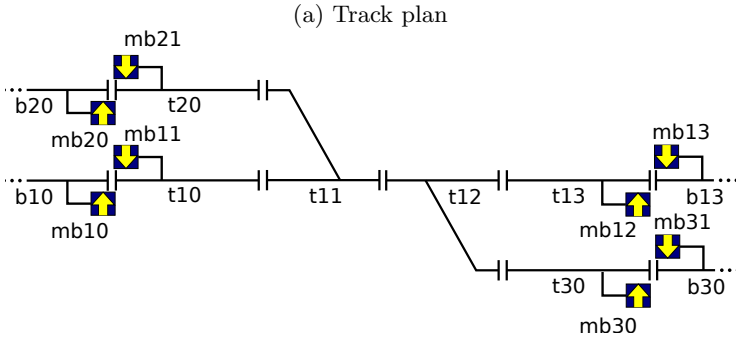
Organisation of the paper. First, in Section 2, we introduce basic notions of the railway domain, including the notions of scheme plan, track plan, and control table. Then, in Section 3, we provide a descriptive comparison between the two different modelling and verification approaches by DTU/Bremen and Surrey/Swansea, where both approaches are described using a similar scheme. In Section 4 we identify a number of error types that might happen during the design of a control table. Finally, in Section 5, we report on experimental results demonstrating that all these errors can be detected by both formal methods, the one from DTU/Bremen as well as the one from Surrey/Swansea.

2 Railway scheme plans

A railway scheme plan consists of a track plan and various tables, e.g., control tables.

Track plans. A railway network consists of a number of track-side elements of different types, for instance linear sections, points, and either marker boards (for ETCS level 2 systems) or physical signals (for legacy systems). The track plan in Fig. 2 shows an example layout of a railway network having six linear sections (b20, t20, b10, t10, t13, b13, t23, b23), two points (t11, t12), and eight marker boards (mb10, ..., mb30). A linear section is a section with up to two neighbours. A point can have up to three neighbours: one at the stem, one at the plus end, and one at the minus end, e.g., point t12 in Fig. 2 has t11, t13, and t30 as neighbours at its stem, plus, and minus ends, respectively. Linear sections and points are collectively called detection sections, as they are used by interlocking systems to detect the presence of trains in a railway network. A point can be switched between two positions: PLUS and MINUS. When it is in the PLUS (MINUS) position, traffic can run from its stem to its plus (minus) end and vice versa. A marker board is installed along a section, and it is used as reference location for an intended travel direction that it is facing, e.g. mb20 in Fig. 2 is installed along section b20, and it is intended for travel direction towards t20. Contrary to legacy systems, in ETCS Level 2, there are no physical signals, but virtual signals associated with marker boards. A virtual signal can be OPEN or CLOSED, respectively, allowing or disallowing traffic to pass the associated marker board. For simplicity, the terms virtual signals, signals, and

(a) Track plan



(b) Control table

Route name	From	To	Route path	Point positions
r1	mb10	mb12	t10;t11;t12;t13	t11:p;t12:p
r2	mb10	mb30	t10;t11;t12;t30	t11:p;t12:m
r3	mb13	mb11	t13;t12;t11;t10	t11:p;t12:p
r4	mb13	mb21	t13;t12;t11;t20	t11:m;t12:p
r5	mb20	mb12	t20;t11;t12;t13	t11:m;t12:p
r6	mb20	mb30	t20;t11;t12;t30	t11:m;t12:m
r7	mb31	mb11	t30;t12;t11;t10	t11:p;t12:m
r8	mb31	mb21	t30;t12;t11;t20	t11:m;t12:m

Fig. 2. Scheme plan "Twist".

marker boards are used interchangeably throughout this paper. Our approach can be used for both, systems using marker boards and classical systems having physical signals.

Control tables. An interlocking system monitors constantly the status of track-side elements, and sets them to appropriate states in order to allow trains travelling safely through the given railway network. A control table specifies the routes in the given network layout and the conditions for setting these routes. A route is a path from a source signal to a destination signal.

In railway signalling terminology, setting a route denotes the process of allocating the resources – i.e. sections, points, signals – for the route, and then locking it exclusively for only one train when the resources are allocated. The specification of a route and conditions for setting and releasing it includes the following information – c.f. the control table shown in Fig. 2: the name of the route (e.g. r1), at which marker board it starts (e.g. mb10) and ends (e.g., mb12), a list of the detection sections in the routes path (e.g. t10;t11;t12;t13) and the required positions of points used by the route (e.g. t11:p;t12:p). Here p and m stands for PLUS and MINUS, respectively.

Note that – for the sake of comparison – we restrict the control table format as illustrated in Fig. 2 to those parts common to both modelling and verification approaches to be discussed in this paper. In general, the DTU/Bremen approach utilises an extended control table discussed in Section 3.2. In contrast, the Sur-

rey/Swansea approach includes release tables which will be described in Section 3.3. In Section 5 on error injection we will discuss how the two different approaches deal with errors injected in the common part. Additionally we will also demonstrate the effect of errors in the extended control tables and in release tables within the DTU/Bremen approach and the Surrey/Swansea approach, respectively.

In order to prevent collision and derailment of trains, route-based interlocking systems employ a basic principle: a route is locked exclusively for use of one train at a time.

In this setting, we consider three safety properties:

1. **collision-freedom** excludes two trains occupying the same track;
2. **run-through-freedom** says that whenever a train enters a point, the point is set to cater for this; e.g., when a train travels from track t20 to track t11, point t11 is set so that it connects t20 and t12 (and not t10 and t12);
3. **no-derailment** says that whenever a train occupies a point, the point does not move.

The correct design for the control table is safety-critical: mistakes can lead to a violation of any of the three safety properties and thus lead to death or serious injury to people, or loss or severe damage to equipment.

3 A descriptive comparison between the modelling and verification approaches of DTU/Bremen and Surrey/Swansea

3.1 Commonalities of both approaches

The DTU/Bremen and the Surrey/Swansea approach share as common starting points:

Assumption We assume track equipment (signals, points, track circuits) to function without mistake, i.e., both our modelling and verification approaches target normal operation.

Narrative As reference point for our modelling we take standard literature on railway signalling such as the book by Kerr and Rowbothan [18], interact with industry, and discuss our models with railway engineers. Communication with practitioners is essential, as the literature is written in jargon and not always as concise as one would wish for as the following quote might illustrate: "When a valid route is received, the interlocking first checks the availability of each set of points in the route and overlap. Points are deemed to be available if they are already lying the correct way, or are not locked the other way." [18].

3.2 DTU/Bremen specialities

This section gives an overview of the verification framework developed by DTU/-Bremen as part of the RobustRailS research project[4]. For details of this framework, see [27, 26, 25, 5].

Overall objectives The framework provides support for an automated 3 step verification and testing approach: (1) First a *static check* is performed on the input scheme plan, (2) then a formal, behavioural system model is automatically generated and *model checked*, (3) and finally *model based testing* of the implemented system is done using test cases, test oracles etc. automatically generated from the formal model. The static check is able to catch errors in scheme plans and in particular in the control tables. The model checking is used to check that the system model is safe and can be used to catch errors in the designed control algorithms as well as to catch errors in the control table if these have not already been found by the static checker. The reason for having the extra static check is to catch as many errors as possible before the more time consuming model checking. The testing is used to catch errors in the *implemented* system. Below we will provide some more details of the two first steps, but not on the testing as that is outside the scope of this paper.

DSL specification of scheme plans The CAD plan and the control table are represented in a DSL. The DTU/Bremen DSL, called Interlocking Configuration Language (ICL), has been formally specified in RSL, and an XML representation has been implemented. An ICL representation can be created manually, or exported from computer-aided design tools supporting the XML format. Alternatively, the user can use the graphical user interface [5] to specify the scheme plan by drawing the track plan and type in the control table via an editor implemented as an Eclipse plug-in. The editor can then export the specification to the XML format. As an option the user may not explicitly provide a control table, but only a track plan and then get a complete control table created automatically from the track plan.

Static check of scheme plans A static checker validates that the track plan and control tables are well-formed, and in case there are errors, it suggests what might be wrong and in some cases also how this can be fixed. The checker validates for instance that any route path in the control table is a connected path in the track plan and that required point positions are correct. It is out of the scope of this paper to list all kinds of checks as there are about 55 of them.

Specification of generic system models For each product family of interlocking systems, a second input is needed: a formal, generic system model. This is given in Interlocking Dynamic Language (IDL), which is another DSL, specially designed for the DTU/Bremen framework. Specifications in this language are similar to RSL-SAL transition system specifications (Kripke model representations consisting of variable declarations and state transition

[4] http://www.imm.dtu.dk/ aeha/RobustRailS/index/

rules) with some additional built-in types and operators. State transition rules for different kind of entities, e.g. the interlocking system controller and track side elements, are placed in different modules and combined by non-deterministic choice (possibly including a prioritization of the transition rules) at the top level.

Automated creation of instantiated system models A system model is automatically created by instantiating the generic IDL system model with data from the ICL scheme plan. Hence, an instantiated model does not include a scheme plan. The resulting instantiated IDL system model is automatically converted to the internal model representation of the RT-Tester tool [21, 24].

Here is an example of an IDL transition rule expressing that when the actual state of a virtual signal s differs from its commanded state, the actual aspect of the signal is updated to the commanded aspect:

$$\texttt{s.ACT} \neq \texttt{s.CMD} \longrightarrow \texttt{s.ACT'} = \texttt{s.CMD}$$

Verification properties The verification properties are automatically created by instantiating a description of generic properties with data from the ICL scheme plan. The resulting verification properties are expressed in RT-Tester as invariants in propositional logic over the state variables of the system model.

Example: For the specific model we used for the experiments reported in this paper, there were no explicit train objects. Instead, train behaviour was implicitly modelled via the occupancy status of track detection sections. This was chosen as it captures behaviours corresponding to all possible numbers of trains, each train having an arbitrary length. The train occupancy of a linear track section t is captured by two integer variables $t.D2U$ and $t.U2D$, one for each of the two travel directions (called down-to-up and up-to-down) through the section. If no train is driving on t in direction down-to-up/up-to-down $t.D2U/t.U2D$ is zero. Hence, there is no head-to-head collision on t, if at least one of the two variables is zero, and this can therefore be expressed as the following invariant:

```
t.D2U * t.U2D = 0
```

Verification in step 2

Verification task (what): It is verified that the invariants hold in all reachable system states of the system model instance.

Verification technique (how): Model checking, more specifically by *k-induction using bounded model checking*. If the system model does not satisfy the invariants, counter-examples will be generated. An interface for visualising the counter-examples at the DSL (ICL) level is integrated into the editor in Eclipse, see [5].

Verification tool: The bounded model checker of the RT-Tester tool.

3.3 Surrey/Swansea specialities

This section gives an overview of the verification framework developed by Surrey/Swansea as part of their SafeCap and Ditto research projects[5]. For details of this framework, see [15, 14].

Overall objective is to verify if a control table is safe w.r.t. a track plan.

Architecture All verification shall be performed by a model checker. This means that also checks that could be performed by some static analyser are encoded as model checking problems. These checks concern conditions on well-formedness of the tables. These lead to two further safety properties, namely:

- "no train on a route with a green signal" – this encodes the check that the route path of the clear table covers all detection sections between two marker boards; and
- "no deviation from the designated route" – this encodes the check that all points on a route path are in the right position to guide the train from the start marker board to the end marker board of a route.

Specification Language for the system model is CSP||B [22], a combination of the process algebra CSP and the B specification language that allows for a combination of event-based and state-based modelling.

We use event-based modelling to capture state changes, e.g., a train moves from one track to the other is represented as `move.A.B`; we use state-based modelling to represent the rules that guide the behaviour of the interlocking, e.g., the conditions under which a route can be set or cancelled.

Modelling – DSL The CAD plan and the tables are first represented in a Domain Specific Language (DSL) before being encoded in CSP||B. This intermediate step allows to implement the whole modelling process as a model transformation in our tool OnTrack [17].

Modelling – Track-plan, tables are represented in dedicated data types in CSP||B.

Modelling – Entities A speciality of the Surrey/Swansea modelling approach is that it directly represents railway entities as part of the specification, i.e., there is a controller, there is an interlocking, there are trains. This allows to observe these identities in simulations, i.e., one can directly see how the train moves, how the state of the interlocking changes, which route requests come from the controller. Besides being helpful in the validation of the modelling approach, this helps to reflect about the model: Surrey/Swansea have proven a number of theorems on their modelling approach, see e.g. [13].

Modelling – System dynamics Active entities, i.e., entities which are able to initiate a system change, are modelled as CSP processes. These are the controller (who can request or cancel route) and the trains (which can move along the track or remain). The interlocking as a passive component, i.e., it reacts to train movements or controller requests, is modelled in B.

[5] http://www.cs.swan.ac.uk/ csmarkus/ProcessesAndData/ditto

For each event that the controller or the trains initiate, the interlocking updates its status and the track equipment according to the dynamic rules as stated in [18]. E.g., a route can be released provided the entry signal of this route is green, all locks of the route are still there, and there is no train in front of the entry signal.

Encoding of the safety conditions The verification properties are encoded as invariants in the specification language B, see the below code representing when there is a collision between two trains on a detection section `t`.

```
Collision(t) == #(t1,t2).(t1 : TRAIN & t2 : TRAIN
     & t1 /= t2 & t1:dom(pos) & t2:dom(pos)
     & (dom({pos(t1)}) = {t}) & (dom({pos(t2)}) = {t}));
```

Verification Technology Model checking with the ProB tool [19]; the tool checks that the invariant holds in all system states.

4 Error injection

Interlocking applications are developed according to the CENELEC standard EN50128 [3] and to processes prescribed by Railway Authorities. For the UK, Network Rail's *Governance for Railway Investment Projects* (GRIP) provides such a process. The first four GRIP phases define the track plan and routes of the railway to be constructed, while phase five – the detailed design – is contracted to a signalling company such as Siemens Rail Automation, UK, which chooses appropriate track equipment, adds control tables to the track plan, and implements the interlocking. Thus, in such a process the track plan is developed first. Only in a second step signalling engineers enrich the track plan with a control table. It is for exactly this second step, namely for the design of a control table that our paper discusses support in terms of formal methods. As track plan and routes come from earlier phases, actually the control table is the element that needs verification.

Track plans can have a considerable size, comprising of hundreds of track-side elements such as linear sections, points, marker boards. This makes control tables complex due to their sheer size measured in numbers of entries needed in the various columns. The control table of Langley – a station which signalling engineers consider to be a small one – has about 160 entries, c.f. [13]. To guarantee safety, every single entry in the control table needs to be correct, none can be forgotten or wrong. To manually check all these entries is a challenging task with a high error probability. It is for that reason, that at Siemens Rail Automation, UK, there are at least three different people who independently perform these checks.

Signalling engineers are well trained to apply various sets of design rules to systematically develop control tables. Thus, one can assume that they have the correct design in mind, however, that due to the sheer number of entries to be produced it is likely that they make a mistake due to an oversight. Consequently, in this paper we inject errors of "syntactic type" into an originally correct control

table, i.e., a control table which we have proven to be safe w.r.t. a given track plan. More precisely we start with a correct control table and apply one of the following error types (ET) to it:

ET1 – leave out one of the track ids in the route path column.
ET2 – exchange one "p" with an "m" or vice verse in the point positions column.
ET3 – delete one point entry in the point positions column.

Then we check if this altered control table still is safe.

Adding elements to the table would not effect safety: as we presume the original control table to be safe anyway, an added element would only further constrain the possible train behaviour – as the original set of train behaviours was already safe, there won't be any safety violations to be found in the reduced set of behaviours.

In the next section on error detection, we will consider various scenarios: we will systematically explore the effects of errors on a number of scheme plans.

5 Error detection

Both, the DTU/Bremen and the Surrey/Swansea approach, verify safety for the given table of "Twist" as shown in Fig. 2 – and for the tables of two more scheme plans "Mini" and "Cross" shown in Figs. 3 and 4 in Sec. 5.4. Thus, starting with a proven to be correct control table, we can perform experiments where errors are injected into the control table.

5.1 Injecting a single error into "Twist"

As the network is double symmetric, i.e., all eight routes are built in the same way, we decided to inject errors only into the row concerning route r1:

Route name	From	To	Route path	Point positions
r1	mb10	mb12	t10;t11;t12;t13	t11:p;t12:p

For route r1 this results in total into eight different errors:

ET1 Four errors $e_1, \ldots e_4$, each by forgetting one track section in the route path.
ET2 Two errors e_5, e_6, each by requiring one point to be in the wrong position.
ET3 Two errors e_7, e_8, each by forgetting a point position.

DTU/Bremen approach For error e_1 – forgetting t10 in the route path column – the static checker provides the error message: `In route r1, two consecutive segments, b10 and t11, are not connected`. Similar outputs are produced for e_2 and e_3. For error e_4 – forgetting the last section, t13, in the route path column such that the route does not end at the exit signal – the error message is: `The exit signal is not placed at the end of the last section of route r1.` For error e_5 – set t11 in wrong position (m rather than

p) – the static checker provides the error message: `For route r1, point t11 is set to MINUS, but it should have been set to PLUS`. Similarly for e_6.

For error e_7 – forgetting to set point t11 – the static checker provides the following error message: `For route r1, point t11 is not given a point position.` Similarly for e_8.

All eight errors have been detected. Note that all errors have been detected by static checking, including a suggestion on what the error might be.

Surrey/Swansea approach For error e_1 – forgetting t10 in the route path column – the ProB tool finds an invariant violation and provides a counter example trace leading to the violating state:

```
request.r1.yes, move.albert.offUnit.b10, nextSignal.albert.b10.green,
move.albert.b10.t10, request.r1.yes, release.r1.yes, request.r4.yes,
move.albert.t10.nullUnit, run-through
```

I.e., in step 6 it is possible to release route r1 although train `albert` is currently on track `t10`; this allows it to set route r4, which moves point t11 to minus; thus, in step 7, train `albert` moves onto a point set in the wrong direction. The tool detects a `run-through`. Similar counter example traces are found for e_2, e_3, e_4, – where a collision is detected.

For error e_5 – set t11 in wrong position (m rather than p) – the ProB tool finds an invariant violation and provides as last step of the counter example trace the event `move.albert.t10.nullUnit`, indicating that train `albert` enters point t11 at the plus end while t11 is connecting the minus end. Also, for error e_6 – set t12 in wrong position (m rather than p) – the ProB tool finds an invariant violation and provides as last two steps of the counter example trace the events `move.bertie.offUnit.b30; move.albert.t30.b30` – i.e., the two trains `bertie` and `albert` collide on section b30.

Similarly, for e_7 – forgetting to set point t11 – ProB comes up with a counter example trace where train `albert` enters the point t11 at the plus end while t11 is connecting the minus end. For e_8 – forgetting to set point t12 – ProB comes up with a counter example trace ending with a collision section b30.

All eight errors have been detected. Note that all errors have been uncovered by model checking where the counter example trace provides an insight into the nature of the first fault detected. Note that further faults might be possible – the tool provides just the first counter example trace found during state exploration.

5.2 Injecting multiple errors in "Twist"

Naturally, it is also possible to inject several errors into one table. Therefore, as in good testing practice, we experiment with at least one scenario including two mistakes at the same time, namely

ET2 – e_6 – set t12 in wrong position (m) and
ET3 – e_7 – forget to set point t11.

In the *DTU/Bremen approach* the static checker find both errors:
`For route r1, point t11 is not given a point position.`
`For route r1, point t12 is set to MINUS, but it should have been set to PLUS.`

In the *Surrey/Swansea approach* the model checker provides a counter example trace classifying the safety violation as a `run-through`.

5.3 Further errors in the parts different in both modellings

As discussed earlier, the control table used above is common to both approaches. However, thanks to national differences and also different suppliers, they both have a richer input format. In the following we experiment with errors outside of the shared input.

DTU/Bremen approach The DTU/Bremen approach has an extended control table that also includes a list of conflicting routes. In case one forgets to include route r2 in the list of conflicting routes for route r1, the static checker highlights this with the message:
`Routes r1 and r2 are in conflict, but route r2 is not listed in the conflicts of route r1. Reasons to be in conflict: Non-concatenated routes with shared elements: t10, t11, t12.`

Yet another column consists of protecting signals that must be closed when setting a route. When one forgets mb11 (on track t10) in the list of protecting signals for route r1, the static checker flags this with the message:
`For route r1, signal mb11 at section t10 should have been listed as a protecting signal.`

Another type of error is to forget to set a point that should provide flank protection for the route. Such errors will be caught by the static checker and suggestions for fixing that will be presented. However, this type of error can't be illustrated for the twist network as there are no such cases.

Surrey/Swansea approach The Surrey/Swansea approach has as a further input a collection of so-called release tables. These tables determine when the locks on points can be released.

One interesting case is when one releases point t11 too early and forgets to include t11 into the route path. Here, the model checker finds the following counter example trace:

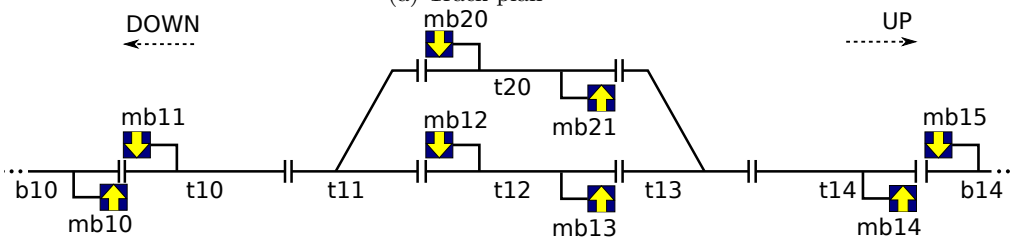
```
move.bertie.offUnit.b10, request.r1.yes nextSignal.bertie.b10.green,
move.bertie.b10.t10, move.albert.offUnit.b10, move.bertie.t10.t11,
request.r1,yes, nextSignal.albert.b10.green, move.albert.b10.t10,
move.albert.t10.t11, collide
```

The trains `albert` and `bertie` collide on point `t11`. This happens as route r1 is wrongly set in step 7: as t11 is not in the route path, the interlocking does not check if this track is free; as t11 is released early, there is also no lock on the point t11.

5.4 Error injection in further scheme plans

In this section we consider error injections in the tables of the two scheme plans shown in Figs. 3 and 4.

(a) Track plan



(b) Control Table

Route name	From	To	Route path	Point positions
r1b	mb10	mb13	t10;t11;t12	t11:p
r2b	mb10	mb21	t10;t11;t20	t11:m
r3	mb12	mb11	t11;t10	t11:p
r4	mb13	mb14	t13;t14	t13:p
r5b	mb15	mb12	t14;t13;t12	t13:p
r6b	mb15	mb20	t14;t13;t20	t13:m
r7	mb20	mb11	t11;t10	t11:m
r8	mb21	mb14	t13;t14	t13:m

Fig. 3. Scheme plan for "Mini"

Mini. For symmetry reasons it is enough to consider route r1b (an entry route) with five errors:

- three of type ET1 (forget t10, t11, and t12, respectively)
- one of type ET2 (set point t11 in wrong position)
- one of type ET3 (forget to set point t11)

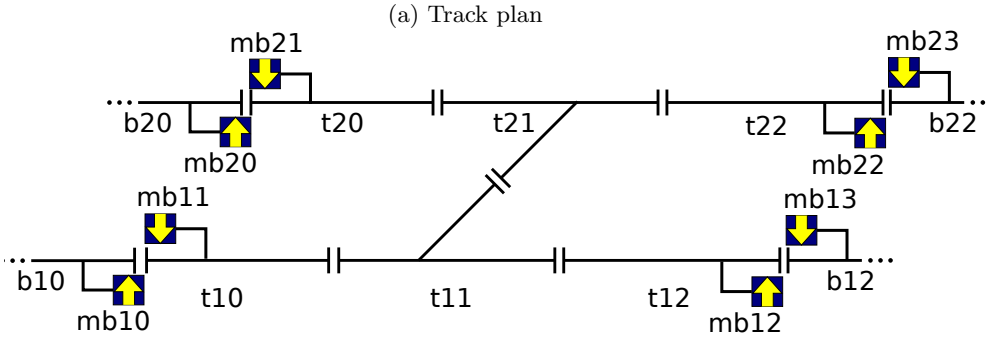
and route r4 (an exit route) with four errors:

- two of type ET1 (forget t13 and t14, respectively)
- one of type ET2 (set point t13 in wrong position)
- one of type ET3 (forget to set point t13)

Cross. For symmetry reasons it is enough to consider routes r1b with five errors:

- three of type ET1 (forget t10, t11, and t12, respectively)
- one of type ET2 (set point t11 in wrong position)

(a) Track plan



(b) Control Table

Route name	From	To	Route path	Point positions
r1b	mb10	mb12	t10;t11;t12	t11:p
r2	mb10	mb22	t10;t11;t21;t22	t11:m;t21:m
r3b	mb13	mb11	t12;t11;t10	t11:p
r4b	mb20	mb22	t20;t21;t22	t21:p
r5b	mb23	mb21	t22;t21;t20	t21:p
r6	mb23	mb11	t22;t21;t11;t10	t11:m;t21:m

Fig. 4. Scheme plan for "Cross"

– one of type ET3 (forget to set point t11)

and route r2 with eight errors:

– four of type ET1 (forget t10, t11, t21 and t22, respectively)
– two of type ET2 (set points t11 and t21 in wrong position, respectively)
– two of type ET3 (forget to set points t11 and t21, respectively)

and route r4b with five errors

– three of type ET1 (forget t20, t21, and t22, respectively)
– one of type ET2 (set point t21 in wrong position)
– one of type ET3 (forget to set point t21)

Results. Both verification approaches find all errors, in case of the DTU/Bremen approach the static checker provides error messages as illustrated above, in case of the Surrey/Swansea approach the ProB model checker finds counter example traces.

6 Summary

This paper presented a first systematic comparison of rail modelling and verification approaches developed by different research groups, in our case the research groups at DTU/Bremen and at Surrey/Swansea.

In order to relate the input of these modelling approaches we defined a common core and discussed differences. In order to relate the formal models, we attempted to present the modelling approaches in a uniform way. In order to relate the verification results we proposed a practicable and useful approach in the form of a testing equivalence: though input and output of both approaches are different, however, both approaches catch the same errors.

For the research community we see this comparison as pioneering work, to which we hope – in the long run – other groups will contribute as well by running their verification approaches through the very same exercises, i.e., this paper can serve as a starting point for a benchmark for railway verification.

In future work we would like

- to consolidate the common input core by including work from further group.
- extend the benchmark with further and larger scheme plans. However, note that in general safety properties of scheme plans are local. This has been exploited by several authors, e.g., for testing and for compositional reasoning [13]. Therefore, one would not expect that considering larger scheme plans would provide new insights into the error detection capabilities of verification approaches.
- to develop a systematic questionnaire for comparing modelling and verification approaches
- to compare actual models by (1) highlighting the different assumptions made for the formal model, i.e., what are the chosen abstractions (e.g., with respect to train length, speed, behaviour of trains, and whether shunting is considered etc.) and (2) highlighting differences in the approaches to route allocation and release (e.g., whether sequential release is used)
- to cover performance aspects such as (1) their scalability, i.e., the limits for the size of track plans and control tables that the approaches can deal with and (2) verification speed, i.e., how long time does it take to verify? For such a comparison to be fair, it needs to be measured for models of the *same* system, but such data is not currently available.

Acknowledgements The DTU/Bremen research has been funded by the RobustRailS project granted by Innovation Fund Denmark. The Surrey/Swansea research has been funded by the SafeCap and the DITTO research projects granted by EPSRC and RSSB. The authors would like to thank Linh Hong Vu for providing the benchmark of scheme plans and the drawings of the track plans.

References

1. M. Banci, A. Fantechi, and S. Gnesi. Some experiences on formal specification of railway interlocking systems using statecharts. In *TRain Workshop at SEFM 2005 (Software Engineering and Formal Methods)*, 2005.
2. Y. Cao, T. Xu, T. Tang, H. Wang, and L. Zhao. Automatic generation and verification of interlocking tables based on domain specific language for computer based interlocking systems. In *CSAE 2011*, pages 511 – 515. IEEE, 2011.

3. C. European Committee for Electrotechnical Standardization. *EN 50128:2011 – Railway applications – Communications, signalling and processing systems – Software for railway control and protection systems*. 2011.
4. A. Ferrari, G. Magnani, D. Grasso, and A. Fantechi. Model Checking Interlocking Control Tables. In E. Schnieder and G. Tarnai, editors, *FORMS/FORMAT 2010*. Springer, 2011.
5. A. Foldager. A graphical domain-specific language for railway interlocking systems. Master's thesis, Technical University of Denmark, DTU Compute, 2015.
6. A. E. Haxthausen. Towards a Framework for Modelling and Verification of Relay Interlocking Systems. In *16th Monterey Workshop: Modelling, Development and Verification of Adaptive Systems: the Grand Challenge for Robust Software*, number 6662 in Lecture Notes in Computer Science, pages 176–192. Springer, 2011.
7. A. E. Haxthausen. Automated Generation of Formal Safety Conditions from Railway Interlocking Tables. *International Journal on Software Tools for Technology Transfer (STTT), Special Issue on Formal Methods for Railway Control Systems*, 16(6):713–726, 2014.
8. A. E. Haxthausen, M. L. Bliguet, and A. A. Kjær. Modelling and Verification of Relay Interlocking Systems. In C. Choppy and O. Sokolsky, editors, *15th Monterey Workshop: Foundations of Computer Software, Future Trends and Techniques for Development*, number 6028 in Lecture Notes in Computer Science. Springer, 2010. Invited paper.
9. A. E. Haxthausen, J. Peleska, and S. Kinder. A Formal Approach for the Construction and Verification of Railway Control Systems. *Formal Aspects of Computing*, 23(2):191–219, 2011. Special issue in Honour of Dines Bjørner and Zhou Chaochen on Occasion of their 70th Birthdays.
10. A. E. Haxthausen, J. Peleska, and R. Pinger. Applied Bounded Model Checking for Interlocking System Designs. In S. Counsell and M. Núñez, editors, *Software Engineering and Formal Methods*, volume 8368 of *Lecture Notes in Computer Science*, pages 205–220. Springer, 2014.
11. A. Iliasov, I. Lopatkin, and A. Romanovsky. Practical formal methods in railways - the safecap approach. In L. George and T. Vardanega, editors, *Reliable Software Technologies - Ada-Europe 2014, 19th Ada-Europe International Conference on Reliable Software Technologies, Paris, France, June 23-27, 2014. Proceedings*, volume 8454 of *Lecture Notes in Computer Science*, pages 177–192. Springer, 2014.
12. P. James, A. Lawrence, M. Roggenbach, and M. Seisenberger. *Towards Safety Analysis of ERTMS/ETCS Level 2 in Real-Time Maude*. Springer. To appear.
13. P. James, F. Moller, N. H. Nga, M. Roggenbach, S. A. Schneider, and H. Treharne. Techniques for modelling and verifying railway interlockings. *STTT*, 16(6):685–711, 2014.
14. P. James, F. Moller, H. N. Nguyen, M. Roggenbach, S. Schneider, and H. Treharne. Decomposing scheme plans to manage verification complexity. *FORMS/FORMAT*, 2014.
15. P. James, F. Moller, H. N. Nguyen, M. Roggenbach, S. A. Schneider, and H. Treharne. On modelling and verifying railway interlockings: Tracking train lengths. *Sci. Comput. Program.*, 96:315–336, 2014.
16. P. James and M. Roggenbach. Encapsulating formal methods within domain specific languages: A solution for verifying railway scheme plans. *Mathematics in Computer Science*, 8(1):11–38, 2014.
17. P. James, M. Trumble, H. Treharne, M. Roggenbach, and S. Schneider. OnTrack: An Open Tooling Environment for Railway Verification. In *NASA Formal Methods*, Lecture Notes in Computer Science. Springer, 2013.

18. D. Kerr and T. Rowbothan. *Introduction to Railway Signalling.* Institution of Railway Signal Engineers, UK, 2001.
19. M. Leuschel, J. Bendisposto, I. Dobrikov, S. Krings, and D. Plagge. *From Animation to Data Validation: The ProB Constraint Solver 10 Years On*, pages 427–446. John Wiley & Sons, Inc., 2014.
20. A. Mirabadi and M. B. Yazdi. Automatic generation and verification of railway interlocking control tables using fsm and nusmv. *Transportation Problems*, pages 103–110, 2009.
21. J. Peleska. Industrial-Strength Model-Based Testing - State of the Art and Current Challenges. In A. K. Petrenko and H. Schlingloff, editors, Proceedings 8th Workshop on Model-Based Testing, Rome, Italy, volume 111 of *Electronic Proceedings in Theoretical Computer Science*, pages 3–28. Open Publishing Association, 2013.
22. S. Schneider and H. Treharne. CSP theorems for communicating B machines. *Formal Asp. Comput.*, 17(4):390–422, 2005.
23. D. Tombs, N. Robinson, and G. Nikandros. Signalling control table generation and verification. In *Proceedings of Cost Efficient Railways through Engineering (CORE 2002)*, pages 415–425. Railway Technical Society of Australasia, 2002.
24. Verified Systems International GmbH. *RT-Tester Model-Based Test Case and Test Data Generator - RTT-MBT - User Manual*, 2013.
25. L. H. Vu. *Formal Development and Verification of Railway Control Systems - In the context of ERTMS/ETCS Level 2.* PhD thesis, 2015.
26. L. H. Vu, A. E. Haxthausen, and J. Peleska. A Domain-Specific Language for Railway Interlocking Systems. In E. Schnieder and G. Tarnai, editors, *FORMS/-FORMAT 2014 - 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems*, pages 200–209. Institute for Traffic Safety and Automation Engineering, Technische Universität Braunschweig, 2014. Got best-paper-award.
27. L. H. Vu, A. E. Haxthausen, and J. Peleska. Formal Modeling and Verification of Interlocking Systems Featuring Sequential Release. In *Formal Techniques for Safety-Critical Systems*, volume 476 of *Communications in Computer and Information Science*, pages 223–238. Springer International Publishing Switzerland, 2015.
28. K. Winter. Optimising Ordering Strategies for Symbolic Model Checking of Railway Interlockings. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies*, volume 7610 of *Lecture Notes in Computer Science*, pages 246–260. Springer, 2012.
29. K. Winter, W. Johnston, P. Robinson, P. Strooper, and L. van den Berg. Tool support for checking railway interlocking designs. In *Proceedings of the 10th Australian workshop on Safety Critical Systems and Software - Volume 55*, SCS '05, pages 101–107, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.
30. Y. T. Yu and M. F. Lau. A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions. *Journal of Systems and Software*, 79(5):577 – 590, 2006. Quality Software.